STUDIA INFORMATICA 2013
Volume 34 Number 2B (112)

Grzegorz DRĄŻEK, Hafed ZGHIDI
Politechnika Śląska, Instytut Informatyki

BADANIE EFEKTYWNOŚCI CZASOWEJ JĘZYKÓW TYPU RPG ORAZ UNIWERSALNEGO JĘZYKA PROGRAMOWANIA DLA DOSTĘPU DO INFORMACJI

Streszczenie. System AS/400 dostarcza wielu wbudowanych, specyficznych dla tej platformy języków programowania. Jednym z nich jest język RPG3, a w nowszej wersji język RPG4. Praca przedstawia możliwości tych języków i bada ich efektywność, porównując je do języka C. Badania dotyczą dostępu do danych zarówno zapisanych w plikach systemowych, jak i danych przechowywanych w systemie bazy danych DB2.

Słowa kluczowe: AS/400, RPG3, RPG4, C, DB2

TIME EFFECTIVENESS OF PROGRAMMING LANGUAGES AND DATA ACCESS METHODS IBM AS/400 SYSTEM

Summary. AS/400 systems offers a lot of built-in (often specific) programming languages. One of them is RPG3 (later RPG4 or ILE RPG). The paper presents the capabilities of these languages and examines their effectiveness by comparing it to the C language. Research focuses on access to data stored in system files and data stored in DB2 database

Keywords: AS/400, RPG3, RPG4, C, DB2

1. Wstęp

Platforma IBM AS/400 (później, iSeries, i5) jest jedną z najbardziej popularnych platform serwerowych, wykorzystywanych w środowiskach biznesowych i produkcyjnych. Zapewnia ona bardzo dużą efektywność w zadaniach interaktywnych, umożliwiając równoczesną obsługę dużej liczby użytkowników. Jednym z ważnych zadań w tych zastosowaniach jest generowanie dużej liczby raportów. Do realizacji tego zadania można wykorzystać specyficzny

dla tej maszyny język programowania RPG (Report Program Generator). Jest to jeden z dostępnych, wbudowanych, dostarczonych w standardzie wraz z maszyną język programowania. Język RPG (początkowo w wersji RPG3, następnie RPG4) nie jest jedynym takim narzędziem, służącym do realizacji tego zadania. Platforma AS/400 dostarcza wielu innych popularnych narzędzi programowych takich, jak: C, Java, PHP.

Celem pracy jest badanie efektywności dostępnych narzędzi programowych na platformie AS/400. Badania mają dwa aspekty: porównanie efektywności czasowej języków RPG3, RPG4 i C w dostępie do danych zapisanych w plikach systemowych oraz sprawdzanie efektywności języka C w dostępie do danych zapisanych w plikach systemowych oraz plikach bazy danych DB2.

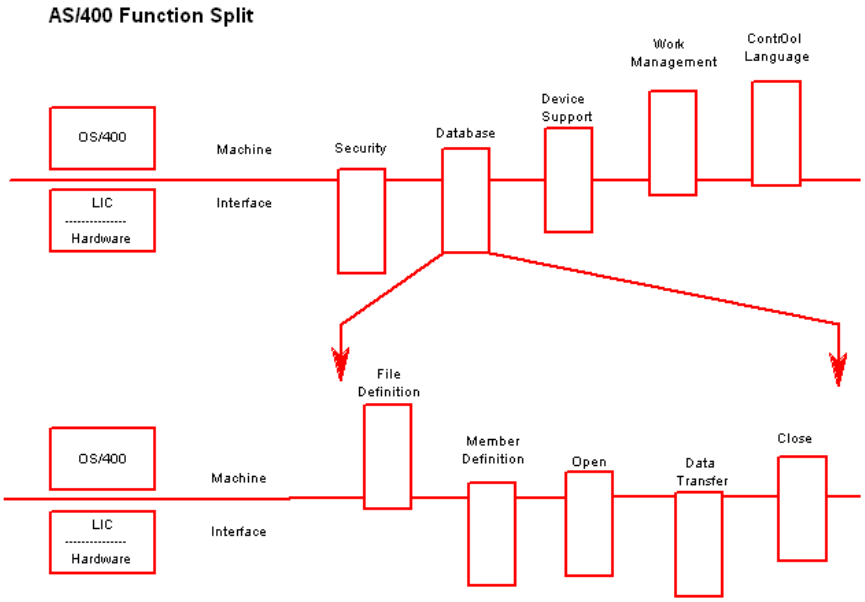
2. Platforma IBM AS/400

Komputery typu IBM AS/400 oferują wiele rozwiązań technologicznych, odmiennych od tych zastosowanych w popularnych systemach komputerowych. Cechy charakterystyczne systemu to:

- Warstwowa architektura maszyny, która uniezależnia użytkownika od sprzętu i umożliwia swobodne przechodzenie do nowszych technologii, bez konieczności ingerencji w już istniejące aplikacje.
- Pamięć jednopoziomowa (ang. Single Level Store). Pamięć główna i pamięć dyskowa widziane są jako ciągła przestrzeń. Dostęp do obiektu przechowywanego w systemie odbywa się poprzez mechanizm adresowania niezależny od fizycznych urządzeń, co oznacza, że dodatkowa pamięć główna lub dyskowa może być dodana do systemu i używana bez wpływu na istniejące programy.
- Orientacja obiektowa. Wszystkie elementy przechowywane w systemie stanowią obiekty, co uniezależnia użytkownika od wewnętrznej struktury maszyny.
- Hierarchia mikroprocesorów – oprócz głównego procesora systemowego, i5/OS ma dużą liczbę innych procesorów.
- System operacyjny OS/400 jako spójna całość. System OS/400 całkowicie integruje wszystkie elementy oprogramowania, wymagane dla większości zastosowań komercyjnych. W OS/400 zintegrowane zostały wszystkie elementy niezbędne dla środowiska aplikacji komercyjnych: relacyjna baza danych (DB2), wspomaganie funkcji komunikacyjnych i sieciowych, mechanizmy bezpieczeństwa i ochrony zasobów oraz wiele innych. Integracja tych elementów w innych systemach wiąże się z trudnościami wynikającymi ze zróżnicowanego pochodzenia komponentów, niezgodności ich wersji, konserwacji itp. W prze-

ciwieństwie do tej sytuacji, w przypadku OS/400 wszystkie elementy stanowią spójną całość. Użytkownik ma dostęp do wszystkich funkcji systemu poprzez jednolity, przyjazny interfejs języka komend Control Language (CL). OS/400 zapewnia również jednolite środowisko dla rozwoju aplikacji.

- Jedną z najbardziej charakterystycznych cech systemu AS/400 jest niezależność warstwy programowej od warstwy sprzętowej. Pełną niezależność od sprzętu uzyskano w wyniku opracowania interfejsu TIMI (ang. Technology Independent Machine Interface). Warstwa ta nie jest związana z żadnym rozwiązaniem sprzętowym. Jest warstwą logiczną, a nie fizyczną i dostarcza programiście i jego aplikacjom sposoby widzenia różnych zasobów systemowych, oddzielając spojrzenie na nie od ich fizycznej realizacji. Sprzęt oraz część systemu, która z tym sprzętem musi komunikować się w sposób bezpośredni, są ukryte pod warstwą TIMI. W ten sposób uzyskano zupełną niezależność od fizycznej architektury procesora oraz innych używanych rozwiązań sprzętowych. Wszystkie elementy systemu operacyjnego AS/400 zależne sprzętowo zostały zaimplementowane jako kod licencjonowany – LIC (Licensed Internal Code). Kod ten zwany jest również SLIC (System Licensed Internal Code) w systemach AS/400, zbudowanych na procesorze RISC. Elementy systemu operacyjnego, niezależne od technologii sprzętowej (a więc zaimplementowane powyżej TIMI), tworzą program OS/400. Każda z funkcji systemu operacyjnego AS/400 zaimplementowana jest częściowo w OS/400 i częściowo w LIC/SLIC. Podział ten jest wynikiem stopnia zależności sprzętowej danej funkcji. Przykładem zróżnicowania mogą być funkcje bazy danych: transfer danych, związany z fizycznym dostępem do dysków, zaimplementowany jest głównie w kodzie LIC/SLIC, zaś definiowanie plików głównie w OS/400. Podział jest przedstawiony na rysunku 1.



Rys. 1. Warstwy systemu AS/400
Fig. 1. AS/400 system layers

3. Baza danych DB2

DB2 Universal Database for AS/400 jest to system zarządzania relacyjną bazą danych, w pełni zintegrowany z AS/400. Z tego powodu baza danych DB2 jest bardzo łatwa w obsłudze i zarządzaniu. DB2 UDB for AS/400 zapewnia szybki dostęp do danych ze względu na to, że większość jej mechanizmów została zaimplementowana w LIC (rys. 1), więc blisko sprzętu. Jak większość systemów baz danych, DB2 udostępnia wiele opcji i funkcji, takich jak: wyzwalacze, procedury przechowywane i dynamiczne bitmapowe indeksy.

Jako interfejs do DB2 w systemie AS/400 mogą służyć takie narzędzia, jak: DB2 Query Manager i SQL Development Kit for AS/400. Stanowią one interaktywne interfejsy do generowania zapytań i raportów. Istnieją również dedykowane prekompilatory i inne narzędzia ułatwiające zanurzenie zapytań SQL w aplikacjach napisanych w językach wyższego rzędu. Implementacja SQL dla systemu operacyjnego OS/400 pozwala, zgodnie ze standardami języka SQL, zdefiniować, manipulować i kontrolować dane oraz dostępem do danych. Dostęp może być zarówno dla danych zapisanych zgodnie z relacyjnym modelem baz danych, jak i dla plików danych systemu AS/400.

4. Języki programowania

Spośród wielu dostępnych języków programowania na maszynie AS/400 (C, C++, CL, Java, RPG3, RPG4), przedstawiony zostanie język programowania RPG. Jest on mało znany ze względu na fakt, że stosowany jest tylko na maszynach AS/400.

W kolejnych rozdziałach zostaną również omówione różnice pomiędzy RPG3 a jego następcą RPG4.

4.1. RPG 3

RPG3 (Report Program Generator) powstał we wczesnych latach 60. XX wieku dla generowania raportów, jako jedno z podstawowych zadań systemu. Nazwy RPG 3, RPG III lub RPG/400 używane są naprzemiennie, w odróżnieniu od RPG 4 lub RPG ILE. Zagadnienie to zostanie omówione w dalszej części pracy.

Proces pisania programów w systemie AS/400 wymaga pewnych wstępnych przygotowań, tj. utworzenia pliku źródłowego i podzbioru (member). Po wykonaniu tych czynności możliwe jest umieszczenie kodu programu i jego kompilacji. W pracy ograniczono się do omawiania najważniejszych elementów programu RPG. Obszerny opis znajduje się w dostępnej literaturze IBM Redbooks.

4.1.1. Pliki źródłowe

Jak już wspomniano, do napisania programu w języku RPG3 konieczne jest utworzenie pliku źródłowego. Do tego celu służy komenda języka CL (Control Language): *CRTSRCPF FILE (biblioteka/ nazwa pliku)*, gdzie *biblioteka* to nazwa biblioteki, w której plik źródłowy ma być tworzony. Biblioteka ta musi oczywiście istnieć w systemie.

Plik źródłowy może być również tworzony z wykorzystaniem odpowiedniej opcji z menu systemowego. Taki plik może zawierać wiele podzbiorów (member), które mogą zawierać kody źródłowe programu.

4.1.2. Podzbiór (member)

Kolejnym krokiem jest utworzenie podzbioru (ang. member) w pliku źródłowym i służy do tego komenda: *STRSEU SRCFILE (biblioteka/nazwa_pliku) SRCMBR (nazwa_podzbioru) TYP(RPG)*, gdzie *biblioteka* to nazwa biblioteki, w której znajduje się plik źródłowy, *nazwa_pliku* to nazwa pliku źródłowego, a TYP (w tym przykładzie RPG) oznacza, w jakim języku programowania member jest napisany (C, Java/ C++, CL…).

Innym sposobem utworzenia podzbioru jest wykorzystanie menu systemowego, wybierając opcję 5 (Programming), a następnie opcję 1 (Programmer menu). Utworzenie podzbioru realizuje się poprzez wybór opcji 8 (Edit a source file member).

Trzecią metodą, pozwalającą na utworzenia podzbioru, jest wykorzystanie menadżera programisty. Do uruchomienia menadżera służy komenda: STRPDM. Następnie należy wybrać opcję 3 (Work with members), podając jako parametry nazwę pliku źródłowego i nazwę biblioteki. Ostatnim krokiem jest wykonanie polecenia utworzenia (Create) poprzez klawisz F6.

4.1.3. Specyfikacja linii kodu

Program RPG zazwyczaj zaczyna się od definicji specyfikacji plików, wymieniając wszystkie pliki do zapisu, odczytu lub modyfikacji. Następnie występuje sekcja deklaracji danych, zawierająca elementy programu, takie jak: struktury danych, tablice, zmienne. Następnie znajduje się specyfikacja sekcji obliczeniowej, która zawiera algorytm realizujący nasze zadanie.

Kod programu składa się z siedmiu różnych rodzajów specyfikacji linii kodu:

- F – definicja plików,
- I – definicja danych wejściowych,
- C – obliczenia,
- O – definicja danych wyjściowych,
- E – przetwarzanie sytuacji rozszerzonych,

- H – kontrolna,
- L – licznik linii.

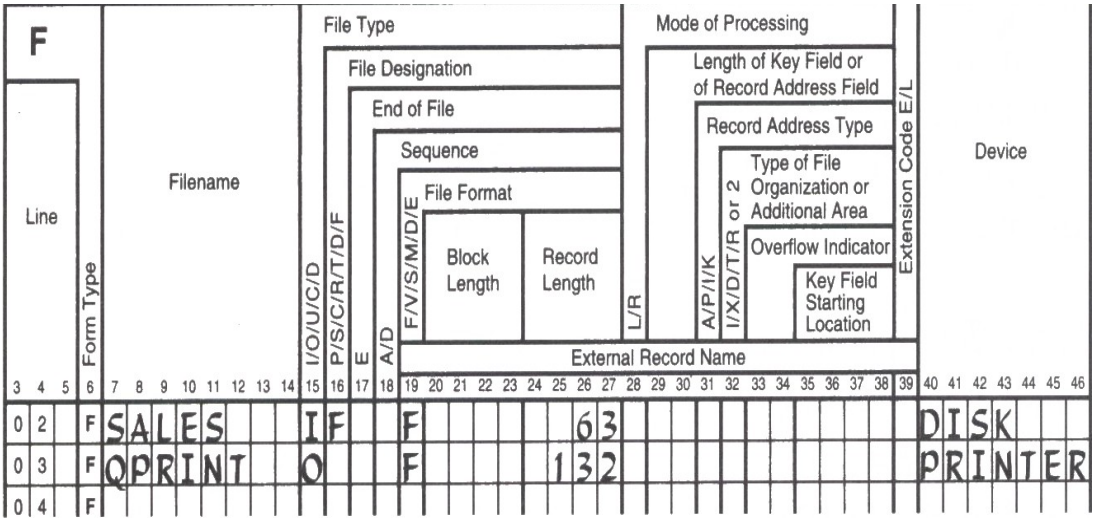
Specyfikacja linii kodu musi się znajdować na pozycji szóstej, np.:

.....**F**FilenameIPEAF.....L..I.....Device+.....

********** Beginning of data **********

FSALES IF F 63 DISK

Na rysunku 2 znajdują się przykłady definicji plików wraz z nagłówkiem.



Rys. 2. RPG3 definicja plików
Fig. 2. RPG3 file definition

4.2. RPG 4

RPG4 jest dostępny od 1994 roku, czyli powstał 35 lat później niż RPG3. Został on wydany wraz z systemem operacyjnym OS/400 w wersji V3R1. Nazwy RPG, RPG IV lub RPG ILE używane są naprzemiennie. RPG4 jest kolejną wersją języka RPG, rozszerzającą poprzednią wersję RPG3 oraz implementującą pewne ułatwienia dla programistów. W pracy ograniczamy się do omawiania najważniejszych różnic pomiędzy RPG3 i RPG4, które przedstawiają się następująco:

- wszystkie funkcje dostępne w wersji RPG3 są również dostępne w RPG4. Umożliwia to konwersję programów napisanych w języku RPG3 do wersji RPG4 bez wprowadzenia dużych zmian w kodzie,
- wprowadzenie wyrażeń typu Free-Format. W RPG3 pozycja w linii słów kluczowych jest stała, natomiast w RPG4 można również użyć Free-Format, gdzie nie jest to wymagane,
- w RPG4 dodano wsparcie dla zmiennych typu data i czas,
- w najnowszej wersji (RPG4) pojawiają się nowe typy danych (liczby zmiennoprzecinkowe, Boolean i wskaźniki),

- w RPG4 utworzona została nowa składnia dla instrukcji sterujących IF, DO (łącznie z nawiasami zwiększającymi czytelność).

4.3. Język C

C i C + + to dwa języki programowania obsługiwane w zintegrowanym środowisku językowym (Integrated Language Environment – ILE). ILE, wraz z systemem operacyjnym OS/400, zapewnia szeroki zakres wsparcia dla nowoczesnych języków programowania. ILE stanowi nową jakość w programowaniu na maszynach AS/400 i zapewnia wsparcie dla wielu języków programowania, debugowanie programów i przyjazne środowiska programowe.

Język C pozwala również na zanurzenie poleceń SQL. Deklarację zmiennych należy umieścić pomiędzy poleceniami:

```
EXEC SQL BEGIN DECLARE SECTION;
```

oraz:

```
EXEC SQL END DECLARE SECTION;
```

Wykonanie poleceń SQL należy umieścić pomiędzy EXEC SQL i znakiem ; (średnika). Przykład poniżej przypisuje do zmiennej aktualny czas systemowy:

```
EXEC SQL BEGIN DECLARE SECTION;
char c_TIME[26];
EXEC SQL END DECLARE SECTION;
EXEC SQL
  SET :c_TIME= CHAR(CURRENT_TIMESTAMP);
```

5. Eksperyment

Mając do dyspozycji lokalny serwer AS/400, przeprowadzono badania efektywności trzech języków programowania w dostępie do danych. Zbadano różnice czasowe w dostępie do danych zapisanych w plikach systemowych za pomocą języka RPG3/RPG4 i C. Do tego celu przygotowano program realizujący to samo zadanie w trzech wersjach językowych. Wyniki tego testu pozwolą na zbadanie szybkości działania tych języków. Drugi aspekt badań dotyczył sprawdzania różnic w dostępie do danych zapisanych w plikach systemowych oraz danych, zapisanych w plikach bazy danych DB2. W tym celu zaprojektowano prostą bazę danych zawierającą te same dane co te zapisane w pliku systemowym. Testy przeprowadzono jedynie dla programu opracowanego w języku C.

Jako zadanie opracowano program generujący raport pracy użytkownika w systemie AS/400. Dane pobierane są z systemowego audytu dla zadanego przedziału czasowego i dla wybranego profilu użytkownika. Warunkiem powodzenia tego programu jest włączenie sys-

temowego audytu, co zapewnia generowanie potrzebnych dla działania programu danych. W sytuacji braku włączonego audytu, program informuje o tym użytkownika i wyświetla krótką instrukcję dotyczącą jego włączenia.

Dla wersji wykorzystującej DB2 umieszczono w pliku bazy danych te same dane, co w pliku audytu. Pozwoliło to, na rzetelne porównanie efektywności różnych wersji programu w dostępie do danych.

5.1. Wyniki badań

Jak już wspomniano, opracowano cztery wersje tego samego programu, wykorzystując trzy różne języki programowania. Tabela 1 przedstawia wyniki badań dla programów generujących raport z pracy użytkowników opierających się na zapisach w pliku audytu systemowego. Dla eliminacji przypadkowych wartości powstałych w wyniku zakłóceń w działaniu systemu, jak np. opóźnień wynikających z chwilowego obciążenia serwera lub wzrostu liczby zadań innych użytkowników, testy powtórzono kilkukrotnie. Przedstawione w tabeli wyniki stanowią średnią arytmetyczną uzyskanych czasów.

Tabela 1

Czas dostępu do danych zapisanych w pliku systemowym (ms)

Liczba wpisów	RPG3	RPG4	C
500	90	130	320
1000	170	240	520
5000	790	1120	1700
10000	1510	2230	3110
25000	3690	5530	7390

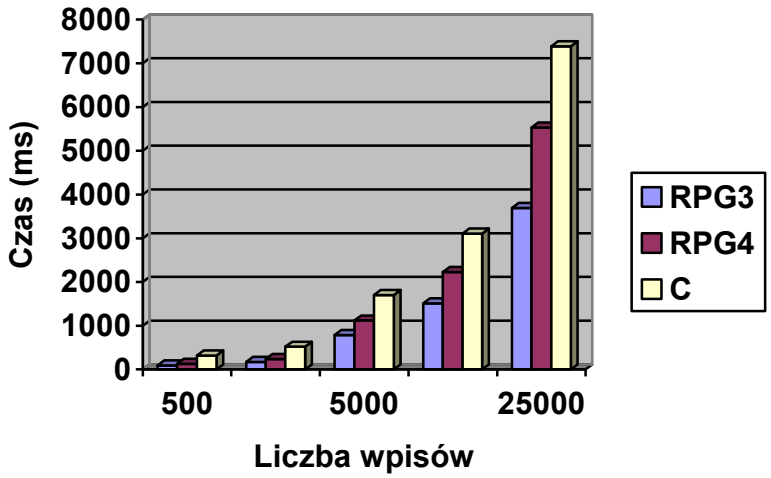
Jak wynika z tabeli, najlepsze wyniki uzyskano dla programu napisanego w języku RPG3, natomiast program napisany w języku C uzyskał najdłuższy czas. Różnice w uzyskanych wynikach są dość znaczne, co dowodzi przystosowania języka RPG3 do tego typu zadań. Język RPG3, mimo dość skomplikowanej i ubogiej składni oraz nieprzyjaznemu pozycyjnemu zapisowi, jest najszybszy właśnie dzięki swojej prostocie. Pojawienie się kolejnej wersji języka w postaci ILE RPG (lub RPG4), z udogodnieniami dla programisty i pojawieniem się dodatkowych składni oraz obsługą innych typów danych, skutkowało wolniejszym kodem wynikowym programów.

W przypadku języka C, który jest językiem wyższego rzędu, niekoniecznie dostosowanym do tego typu zadań (generowania raportów), wyniki są najgorsze. Taki wynik potwierdza oczekiwania, gdyż odwołania do funkcji systemowych i bibliotek obsługujących język skutkują mniejszą wydajnością. Rysunek 3 przedstawia wyniki z tabeli 1.

Warto zauważyć, że czas dostępu do danych nie jest wprost proporcjonalny do liczby wpisów i że stosunek ten jest prawie stały w przypadku RPG3 i RPG4, ale maleje w przypad-

ku C. Zakładając, że dla nas jednostką jest 500 wpisów, dwukrotny wzrost liczby wpisów (1000) powoduje uzyskanie 1,9x dłuższego czasu w przypadku RPG3 i podobną wartość jak w przypadku RPG4, ale stosunek ten jest niższy (1,6) w przypadku C. Zwiększając liczbę wpisów 10-krotnie (5000 wpisów), uzyskuje się odpowiednio około 8,5x dłuższy czas dla programów napisanych w RPG, ale niewiele ponad 5x dłuższy w przypadku C. Podobnie przedstawia się sytuacja w przypadku 20- i 50-krotnie większej liczby wpisów. Uzyskuje się wtedy czasy dłuższe o około odpowiednio 17x i 41x dla RPG oraz około 10x i 23x dla języka C.

Podsumowując, wraz ze zwiększeniem liczby wpisów, różnica w działaniu programów napisanych w językach RPG i C maleje. Tabela 2 i rysunek 4 przedstawiają wynik analizy.



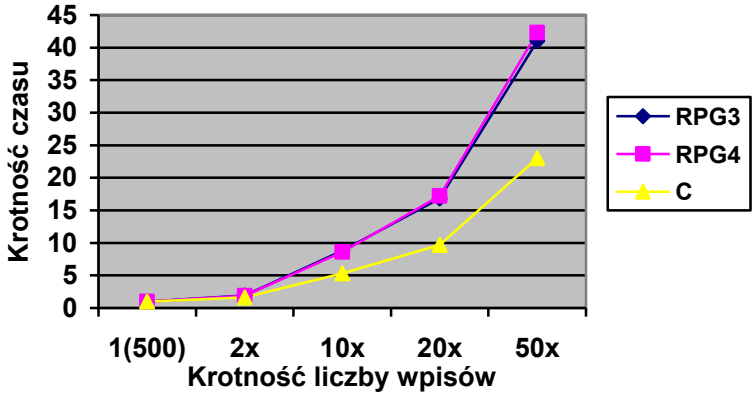
Rys. 3. Czas dostępu do danych
Fig. 3 Data access time

Tabela 2

Stosunek wzrostu czasu dostępu do danych do liczby wpisów

Liczba wpisów (jednostka)	RPG3	RPG4	C
500(1x)	1	1	1
1000(2x)	1,9	1,85	1,6
5000(10x)	8,78	8,6	5,3
10000(20x)	16,8	17,2	9,7
25000(50x)	41	42,3	23

Drugi obszar badań dotyczył zachowania się języka C przy wykorzystaniu systemu bazy danych DB2. Do realizacji tego zadania tworzono prostą bazę danych, zawierającą, tak jak już wspomniano, te same dane (i tyle samo rekordów, ile wpisów) co w pliku dziennika. Uzyskane wyniki przedstawia tabela 3.



Rys. 4. Stosunek wzrostu czasu dostępu do liczby wpisów
Fig. 4 Access time ratio depending on entries number

Tabela 3

Stosunek wzrostu czasu dostępu do danych C/C + DB2

Liczba wpisów/rekordów	C	C + DB2	Stosunek
500	320	290	-10%
1000	520	420	-20%
5000	1700	1640	-4%
10000	3110	3150	+1,01%
25000	7390	7550	+1,02%

Jak wynika z tabeli, dla małej liczby rekordów bazy danych czas dostępu do danych jest dużo krótszy (nawet do 20%) niż w przypadku danych zapisanych w plikach systemowych. Może to być spowodowane tym, że w przypadku małej liczby rekordów wszystkie dane (rekordy) mieszczą się w buforach pamięci systemu OS/400 (ang. Buffer Pool), co minimalizuje liczbę operacji dyskowych. W przypadku dużej liczby rekordów, DB2 odwołuje się częściej do pamięci dyskowej, tak samo jak w przypadku programu operującego bezpośrednio na plikach systemowych. Takie operacje są, jak wiadomo, dużo dłuższe niż operacje wykonywane w pamięci operacyjnej.

6. Podsumowanie

Celem pracy było zbadanie efektywności czasowej rożnych języków programowania w dostępie do danych systemu AS/400. Przedstawiono wyniki badań dla 4 różnych wersji tego samego algorytmu, opracowanego w trzech różnych językach (RPG3, RPG4 i C) i operującego na danych zapisanych w plikach systemowych lub przechowywanych w systemie bazy danych DB2. Zadaniem programu były analiza pracy użytkowników systemu AS/400 i generowanie, na podstawie zapisów pliku systemowego audytu lub rekordów bazy danych, odpowiednich raportów.

Jak się okazało, najlepsze wyniki uzyskano dla programu napisanego w języku RPG3. Jest to język bardzo prosty i efektywny czasowo. Mimo mankamentów, które opisano w pracy, stanowi on bardzo wydajne narzędzie do realizacji tego typu zadań. Ta zaleta tłumaczy jego szerokie zastosowanie do dnia dzisiejszego, mimo że minęły 53 lata od jego powstania.

Pojawienie się kolejnej wersji języka RPG, ILE RPG lub RPG4, z udogodnieniami dla programisty i obsługą większej liczby typów danych, powodowało uzyskanie nieco gorszych wyników, ale lepszych od języka C, który, jak się okazało, nie jest dostosowany do tego typu zadań.

Wykorzystanie bazy danych DB2 pozwoliło na uzyskanie lepszych wyników jedynie w przypadku małej liczby rekordów. W przypadku dużej ilości danych okazało się, że wyniki są porównywalne z wynikami uzyskanymi w programie operującym bezpośrednio na plikach systemowych.

BIBLIOGRAFIA

1. Bozman J. S., Perry R.: Server Cost of Ownership in ERM Customer Sites. A Total Cost of Ownership Study. An IDC White Paper, 2001.
2. Department Of Defense: Trusted Computer System Evaluation Criteria. l985.
3. IBM AS/400 Advanced Series: Application Display Programming, 1st edition. SC41-5715-00, 1997.
4. IBM AS/400: Application Development ToolSet for AS/400 Screen Design Aid, 1st edition. SC09-2604, 1998.
5. IBM AS/400e: AS/400e System Handbook, 20th edition. GA19-5486-19, 2000.
6. IBM AS/400e: CL Reference, 3rd edition. SC41-5722-02, 1998.
7. IBM AS/400 Advanced Series: Communication Configuration, 1st edition. SC41-5401-00, 1997.
8. IBM AS/400e: OS/400 Message Handling APIs, System API Reference, 4th edition. SC41-5862-02.
9. IBM Power Systems advantages, http://www-03.ibm.com/systems/power/advantages/power.html.
10. IBM AS/400 Advanced Series: Remote Workstation Support, 1st edition. SC41-5402-00, 1997.
11. IBM iSeries: Security Reference, 6th edition. SC41-5302-05, 2001.
12. IBM AS/400e: Work Management, 4th edition. SC41-5306-03, 1999.
13. Soltis F. G.: Inside the AS/400, 2nd edition. 29th Street Press, 1997.
14. Power Systems Software, AIX, IBM and Linux software solutions, http://www-03.ibm.com/systems/power/software/.

15. RPG Reference, 1st edition. C0918170, 1994.
16. Application System/400, RPG/400 User's Guide, 1st edition, SC09-1816-00, 1994.
17. Yaeger J.: Programming in RPG/400, 2nd edition. 1995.
18. Showcase Article – Converting RPGIII to RPG-ILE, http://www.texas400.com/SAconvertRPGIIItoILE.html.

Wpłynęło do Redakcji 16 stycznia 2013 r.

Abstract

The aim of this project was to examine the time effectiveness of various programming languages in access to AS/400 system database. There were presented results of the research for 4 different versions of the same algorithm, which was worked out in 3 different languages (RPG3, RPG4 and C), trading in data stored in system files or the ones stored in DB2 database.

The task of the program was the analysis of the users' occupation of AS/400 system and generating on the bases of system files audit or database records proper reports.

As it turned out, the best results received the program written in RPG3 language. It is a very simple and time effective language. Despite drawbacks (as mentioned), it is a very effective instrument allowing realization of such tasks. Despite 53 years since it began to exist, this advantage explains its extended application until today.

Appearing another version of RPG, ILE RPG (or RPG4) language with all conveniences for the programmer as well as, possibility of handling more database types, caused a bit worse results, but better ones than C language, which, as occurred, is not adapted to this type of tasks.

Using DB2 database allowed getting better results only in case of small number of records. In case of significant amount of data, the results were comparable to the ones obtained in program operating directly on system files.

Adresy

Hafed ZGHIDI: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16,
44-100 Gliwice, Polska, hafed.zghidi@polsl.pl.
Grzegorz DRĄŻEK: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16,
44-100 Gliwice, Polska, drazek_g@poczta.onet.pl